

CONCEPTE GEOMETRICE ÎN GRAFICA PE CALCULATOR

Mihai-Sorin Stupariu

Sem. I, 2014-2015

Cuprins

1	Generalități	2
1.1	Exemplu de program OpenGL	2
1.2	Biblioteci utilizate de OpenGL și funcții asociate	3
2	Primitive grafice. Atribute ale primitivelor grafice	4
2.1	Fundamente teoretice	4
2.1.1	Intersecții de segmente	4
2.1.2	Poligoane	4
2.2	Algoritmi de rasterizare pentru segmente	5
2.3	Funcții OpenGL	5
2.3.1	Vârfuri	5
2.3.2	Puncte	6
2.3.3	Segmente de dreaptă	7
2.3.4	Poligoane	8
2.4	Exerciții	11

Capitolul 1

Generalități

1.1 Exemplu de program OpenGL

```
#include <windows.h> // biblioteci care urmeaza sa fie incluse.
#include <gl/freeglut.h> // nu trebuie uitat freeglut.h
void init (void) // initializare ecran de vizualizare
{
    glClearColor (1.0, 1.0, 1.0, 0.0); // culoarea de fond a ecranului
    glMatrixMode (GL_PROJECTION); // reprezentare 2D; proiectie ortogonala
    gluOrtho2D (0.0, 1200.0, 0.0, 700.0);
}
void desen (void) // procedura desenare
{
    glColor3f (0.0, 0.0, 1.0); // culoarea punctelor: albastru
    glBegin (GL_POINTS);
        glVertex2i (20, 20);
        glVertex2i (21, 21);
        glVertex2i (22, 22);
        glVertex2i (23, 23);
        glVertex2i (24, 24);
        glVertex2i (27, 27);
        glVertex2i (100, 100);
    glEnd ( );
    glColor3f (1.0, 0.0, 0.0); // culoarea liniei frante: rosu
    glBegin (GL_LINE_STRIP);
        glVertex2i (0,0);
        glVertex2i (200, 100);
        glVertex2i (300, 120);
        glVertex2i (255, 290);
    glEnd ( );
    glColor3f (0.0, 1.0, 0.0); // culoarea reuniunii de segmente: verde
    glBegin (GL_LINES);
        glVertex2i (400,400);
        glVertex2i (600, 500);
        glVertex2i (700, 520);
        glVertex2i (655, 690);
    glEnd ( );
    glFlush ( );
}
void main (int argc, char** argv)
{
    glutInit (&argc, argv);
    glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
```

```

glutInitWindowPosition (100, 100); // pozitia initiala a ferestrei de vizualizare
glutInitWindowSize (1200, 700); // dimensiunile ferestrei
glutCreateWindow ("Puncte & Segmente"); // titlul ferestrei
init ( );
glClear (GL_COLOR_BUFFER_BIT);
glutDisplayFunc (desen);
glutMainLoop ( );
}

```

1.2 Biblioteci utilizate de OpenGL și funcții asociate

- bibliotecă fundamentală (core library): este independentă de platforma pe care se lucrează; funcțiile corespunzătoare au prefixul `gl` (de exemplu: `glVertex ( )`; `glColor ( )`; `glBegin ( )`, etc.
- GLU (OpenGL Utility): conține mai ales proceduri / funcții legate de proiecție, precum și funcții pentru conice și quadrice; funcțiile asociate au prefixul `glu`
- GLUT (OpenGL Utility Toolkit) / freeglut: bibliotecă *dependentă de sistem*, utilizată pentru a realiza fereastra de vizualizare; poate interacționa cu sisteme de operare bazate pe ferestre de vizualizare; funcțiile specifice au prefixul `glut`

Capitolul 2

Primitive grafice. Atribute ale primitivelor grafice

2.1 Fundamente teoretice

2.1.1 Intersecții de segmente

În plan (context 2D)

- Segmentele $[AB]$ și $[CD]$ se intersectează $\Leftrightarrow A$ și B sunt de o parte și de alta a dreptei CD și C și D sunt de o parte și de alta a dreptei AB .
- Două puncte M și N sunt de o parte și de alta a dreptei d de ecuație $f(x, y) = \alpha x + \beta y + \gamma = 0 \Leftrightarrow f(M) \cdot f(N) < 0$.

În spațiu (context 3D)

- **Varianta 1** Reducere la cazul 2D. Se găsește o izometrie / transformare afină astfel ca figura să fie situată în planul $z = 0$.
- **Varianta 2** Se folosește reprezentarea segmentelor cu ajutorul combinațiilor afine. Segmentele $[AB]$ și $[CD]$ se intersectează $\Leftrightarrow$

$$\exists s_0, t_0 \in [0, 1] \quad \text{a.î.} \quad (1 - t_0)A + t_0B = (1 - s_0)C + s_0D.$$

2.1.2 Poligoane

Reguli referitoare la vârfurile $P_1, P_2, \dots, P_N$ (diferite două câte două) care determină un poligon, pentru ca GL-POLYGON să producă efectul dorit:

1. Punctele trebuie să fie coplanare. **De verificat:** condiția de coplanaritate

$$\text{rang} \begin{pmatrix} 1 & 1 & 1 & \dots & 1 \\ x_{P_1} & x_{P_2} & x_{P_3} & \dots & x_{P_N} \\ y_{P_1} & y_{P_2} & y_{P_3} & \dots & y_{P_N} \\ z_{P_1} & z_{P_2} & z_{P_3} & \dots & z_{P_N} \end{pmatrix} = 3 \quad (2.1)$$

sau faptul că

$$\dim_{\mathbb{R}} \langle \overrightarrow{P_1P_2}, \overrightarrow{P_1P_3}, \dots, \overrightarrow{P_1P_N} \rangle = 2. \quad (2.2)$$

O condiție alternativă este coliniaritatea vectorilor $\overrightarrow{P_1P_2} \times \overrightarrow{P_2P_3}, \overrightarrow{P_2P_3} \times \overrightarrow{P_3P_4}, \dots, \overrightarrow{P_{N-1}P_N} \times \overrightarrow{P_NP_1}, \overrightarrow{P_NP_1} \times \overrightarrow{P_1P_2}$

2. Vârfurile trebuie indicate în ordinea corectă, astfel încât linia poligonală să nu aibă autointersecții. **De verificat:** intersecții de segmente (cf. secțiunea 2.1.1).
3. Poligonul trebuie să fie convex. **De verificat:** convexitatea (folosind produse vectoriale).

2.2 Algoritmi de rasterizare pentru segmente

Fie $M_0 = (x_0, y_0)$, $M_{End} = (x_{End}, y_{End})$ extremitățile unui segment ce urmează să fie reprezentat în format raster (cu $x_0, y_0, x_{End}, y_{End} \in \mathbb{N}$). Se presupune că dreapta M_0M_{End} are ecuația

$$y = mx + n. \quad (2.3)$$

Notății și proprietăți:

$$\Delta x = x_{End} - x_0, \quad \Delta y = y_{End} - y_0, \quad m = \frac{\Delta y}{\Delta x}.$$

Presupunem $\Delta x > \Delta y > 0$. Numărul de pași care trebuie realizați în lungul axei orizontale Ox este mai mare decât numărul de pași care trebuie realizați în lungul axei verticale Oy .

- **Varianta 1** Implementare directă a ecuației (2.3)
- **Varianta 2** Algoritmul DDA
- **Varianta 2** Algoritmul lui Bresenham

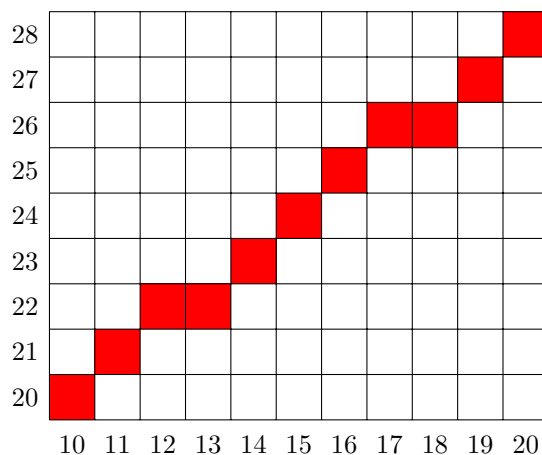


Figura 2.1: **Algoritmul DDA / algoritmul lui Bresenham.**
Pixelii selectați pentru a uni punctele $(10, 20)$ și $(20, 28)$ sunt colorați cu roșu.

2.3 Funcții OpenGL

2.3.1 Vârfuri

Primitivele grafice sunt realizate cu ajutorul **vârfurilor**. În OpenGL un vârf este definit cu ajutorul funcției

```
glVertex* ( );
```

Cu ajutorul sufixului * se indică:

- dimensiunea spațiului în care lucrăm, $n \in \{2, 3, 4\}$
- tipul de date utilizat, care poate fi:
 - **i** (integer)
 - **s** (short)
 - **f** (float)
 - **d** (double)
- (opțional) posibila formă vectorială, indicată prin sufixul **v**.

Unui vârf îi sunt asociate:

- coordonatele (fac parte din definiție),
- culoarea,
- normala (legată de funcții de iluminare),
- coordonate de texturare

O funcție de tipul **glVertex** poate fi apelată într-un cadru de tip

```
glBegin (*);  
  
glEnd;
```

(unde * reprezintă tipul de primitivă generat).

2.3.2 Puncte

Astfel, cu ajutorul acestei funcții pot fi generate puncte, într-un cadru de tipul

```
glBegin (GL_POINTS);  
  
glEnd;
```

segmente de dreaptă (vezi paragraful 2.3.3) sau poligoane.

Exemplul 2.1 Următoarele două secvențe de cod sursă sunt echivalente, având ca efect desenarea punctelor de coordonate (50,100), (70,80) și (90,150).

I.

```
glBegin(GL_POINTS);  
    glVertex2i (50, 100);  
    glVertex2i (70, 80);  
    glVertex2i (90, 150);  
glEnd;
```

II.

```
int p1[ ] = {50, 100};  
int p2[ ] = {70, 80};  
int p3[ ] = {90, 150};
```

```
glBegin(GL_POINTS);
    glVertex2iv (p1);
    glVertex2iv (p2);
    glVertex2iv (p3);
glEnd;
```

Atribute ale punctelor

- Culoarea (dată de culoarea vârfului)
- Dimensiunea

```
glPointSize (dimens);
```

2.3.3 Segmente de dreaptă

Un singur segment de dreaptă, determinat de punctele de coordonate (x_1, y_1) , (x_2, y_2) (vom lua $x_1, x_2, y_1, y_2 \in \mathbf{Z}$) poate fi trasat utilizând următoarea secvență de cod sursă:

```
glBegin (GL_LINES);
    glVertex2i (x1, y1);
    glVertex2i (x2, y2);
glEnd;
```

De asemenea, pentru trasarea segmentelor de dreaptă mai pot fi utilizate și constantele simbolice `GL_LINES`, `GL_LINE_STRIP`, `GL_LINE_LOOP`. Să considerăm o mulțime de puncte având coordonatele (pe care le presupunem numere întregi) $P_1 = (x_1, y_1), P_2 = (x_2, y_2), \dots, P_n = (x_n, y_n)$.

- Putem desena segmentele de forma $[P_{2k+1}P_{2k+2}]$ (în cazul în care n este impar, ultimul punct nu este unit cu nimeni) folosind instrucțiunile

```
glBegin (GL_LINES);
    glVertex2i (x1, y1);
    glVertex2i (x2, y2);
    .....
    glVertex2i (xn, yn);
glEnd;
```

- Linia frântă ce unește punctele $P_1, P_2, \dots, P_n$ este trasată astfel:

```
glBegin (GL_LINE_STRIP);
    glVertex2i (x1, y1);
    glVertex2i (x2, y2);
    .....
    glVertex2i (xn, yn);
glEnd;
```

- Linia poligonală închisă determinată de punctele $P_1, P_2, \dots, P_n$ (poate conține autointersecții!) este desenată folosind comenzile:

```
glBegin (GL_LINE_LOOP);
    glVertex2i (x1, y1);
    glVertex2i (x2, y2);
    .....
    glVertex2i (xn, yn);
glEnd;
```

Atribute ale segmentelor de dreaptă

- Culoarea (dată de culorile extremităților segmentului); când acestea din urmă au culori diferite, culorile punctelor segmentului sunt determinate folosind **interpolarea afină**. Modul de trasare se controlează cu

```
glShadeModel (...);
```

fiind posibile variantele

```
glShadeModel (GL_SMOOTH);
```

```
glShadeModel (GL_FLAT);
```

- Lățimea

```
glLineWidth (width);
```

- Modul de desenare (șablon, model)
Este definit prin

```
glLineStipple(repeatfactor, pattern);
```

și activat / dezactivat prin

```
glEnable (GL_LINE_STIPPLE);
.....
glDisable (GL_LINE_STIPPLE);
```

2.3.4 Poligoane

Un singur triunghi, determinat de punctele de coordonate (x_1, y_1) , (x_2, y_2) , (x_3, y_3) (vom lua $x_1, x_2, y_1, y_2, x_3, y_3 \in \mathbf{Z}$), poate fi trasat utilizând următoarea secvență de cod sursă:

```
glBegin (GL_TRIANGLES);
    glVertex2i (x1, y1);
    glVertex2i (x2, y2);
    glVertex2i (x3, y3);
glEnd;
```

De asemenea, pentru trasarea segmentelor de dreaptă mai pot fi utilizate și constantele simbolice `GL_TRIANGLES`, `GL_TRIANGLE_STRIP`, `GL_TRIANGLE_FAN`, `GL_QUADS`, `GL_QUAD_STRIP`, `GL_POLYGON`. Să considerăm o mulțime de puncte

având coordonatele (pe care le presupunem numere întregi) $P_1 = (x_1, y_1), P_2 = (x_2, y_2), \dots, P_n = (x_n, y_n)$.

- Putem desena triunghiuri de forma $[P_{3k+1}P_{3k+2}P_{3k+3}]$ (în cazul în care n nu este divizibil prin 3, există vârfuri care nu aparțin niciunui triunghi) folosind instrucțiunile

```
glBegin (GL_TRIANGLES);
    glVertex2i (x1, y1);
    glVertex2i (x2, y2);
    .....
    glVertex2i (xn, yn);
glEnd;
```

- Triunghiuri care au muchii în comun, definite de punctele $P_1, P_2, \dots, P_n$ sunt trasate astfel:

```
glBegin (GL_TRIANGLE_STRIP);
    glVertex2i (x1, y1);
    glVertex2i (x2, y2);
    .....
    glVertex2i (xn, yn);
glEnd;
```

- Un evantai de triunghiuri având vârful P_1 comun este desenat folosind comenzile:

```
glBegin (GL_TRIANGLE_FAN);
    glVertex2i (x1, y1);
    glVertex2i (x2, y2);
    .....
    glVertex2i (xn, yn);
glEnd;
```

- Patrulaterale pot fi desenate folosind instrucțiunile

```
glBegin (GL_QUADS);
    glVertex2i (x1, y1);
    glVertex2i (x2, y2);
    .....
    glVertex2i (xn, yn);
glEnd;
```

```
glBegin (GL_QUAD_STRIP);
    glVertex2i (x1, y1);
    glVertex2i (x2, y2);
    .....
    glVertex2i (xn, yn);
glEnd;
```

- Pentru poligoane convexe (vezi secțiunea 2.1.2 pentru condițiile pe care trebuie să le verifice punctele) se poate folosi secveța de cod

```
glBegin (GL_POLYGON);
  glVertex2i (x1, y1);
  glVertex2i (x2, y2);
  .....
  glVertex2i (xn, yn);
glEnd;
```

- Există o funcție specială pentru desenarea unui dreptunghi care are ca vârfuri diagonale opuse punctele (x_1, y_1) și (x_2, y_2) .

```
glRect* (x1,y1,x2,y2)
```

Atribute ale triunghiurilor

- Culoarea (dată de culorile vârfurilor); când acestea din urmă au culori diferite, culorile punctelor segmentului sunt determinate folosind **interpolarea afină**. Modul de trasare se controlează, ca în cazul segmentelor, cu

```
glShadeModel (...);
```

fiind posibile variantele

```
glShadeModel (GL_SMOOTH);
```

```
glShadeModel (GL_FLAT);
```

- Funcții OpenGL pentru trasarea feței / spatelui unui poligon

- Precizarea modului de redare

```
glPolygonMode (face, mode);
```

Pentru *face* sunt posibile valorile GL_FRONT, GL_BACK, GL_FRONT_AND_BACK, iar pentru *mode* sunt posibile valorile GL_POINTS, GL_LINE, GL_FILL. Valoarea implicită este GL_FILL.

- Schimbarea ordinii vârfurilor

```
glFrontFace
(GL_CW);
```

```
glFrontFace(GL_CCW);
```

Valoarea implicită este GL_CCW și revenirea la aceasta trebuie precizată în mod explicit.

- Renunțare la trasarea feței unor poligoane

Poligoanele văzute dinspre fața *face* pot fi eliminate cu secvența

```
glEnable (GL_CULL_FACE);
glCullFace (face);
..... listă poligoane convexe
glDisable (GL_CULL_FACE);
```

- Șablonul de desenare

2.4 Exerciții

Exercițiul 2.4.1 Fie $A = (3, 1)$, $B = (5, 3)$, $C = (10, 8)$, $D = (1, -1)$. Stabiliți dacă segmentele $[AB]$ și $[CD]$ se intersectează și, în caz afirmativ, determinați coordonatele punctului de intersecție.

Exercițiul 2.4.2 Fie $A = (1, 2, 3)$, $B = (3, 2, 5)$, $C = (8, 6, 2)$, $D = (-1, 0, 3)$. Stabiliți dacă segmentele $[AB]$ și $[CD]$ se intersectează și, în caz afirmativ, determinați coordonatele punctului de intersecție.

Exercițiul 2.4.3

- a) Verificați echivalența dintre condițiile (2.1) și (2.2).
- b) Care este interpretarea geometrică dacă egalitatea din condiția (2.1) devine inegalitate?

Exercițiul 2.4.4 Calculați produsul vectorial $v \times w$, pentru $v = (1, 2, 0)$ și $w = (-3, 1, 0)$. Aceeași cerință pentru $v = (2, 4, 1)$ și $w = (-1, -1, 1)$.

Exercițiul 2.4.5 Stabiliți natura virajelor din poligonul $P_1P_2P_3P_4P_5P_6$, pentru $P_1 = (1, 0)$, $P_2 = (4, 0)$, $P_3 = (7, 2)$, $P_4 = (5, 4)$, $P_5 = (8, 6)$, $P_6 = (0, 7)$.

Exercițiul 2.4.6 Considerăm punctele $P_1 = (2, 3, 5)$, $P_2 = (3, 4, 6)$, $P_3 = (0, 3, 4)$, $P_4 = (3, 2, 5)$. Stabiliți care este poziția lui P_4 față de $\Delta P_1P_2P_3$.

Exercițiul 2.4.7 Fie P_1, P_2, P_3 trei puncte necoliniare din $\mathbb{R}^3$. Considerăm ecuația planului $Ax + By + Cz + D = 0$ obținută prin dezvoltarea determinantului din ecuația (2.1). Verificați că $\overrightarrow{P_1P_2} \times \overrightarrow{P_2P_3} = (A, B, C)$.

Exercițiul 2.4.8 Fie punctele $A = (2, 3, 0)$, $B = (5, 4, 0)$, $C = (0, 8, 0)$. Stabiliți ordinea în care trebuie indicate punctele astfel ca punctul $(-1, 2, 6)$ să fie în fața planului triunghiului determinat de ele.

Exercițiul 2.4.9 Considerăm punctele $A = (-1, 0, 1)$, $B = (1, 0, 1)$, $C = (1, 0, -1)$, $D = (-1, -0, -1)$. În ce ordine trebuie indicate punctele pentru ca fața poligonului să fie orientată spre prelungirea axei Oy ?

Exercițiul 2.4.10 La reprezentarea unui segment folosind algoritmul DDA sunt selectați 11 pixeli (inclusiv cei corespunzând extremităților segmentului). Calculați panta dreptei suport a segmentului, știind că dintre pixelii selectați doar doi au aceeași ordonată. Dați exemplu de puncte M_0 și M_{End} care să verifice aceste condiții și scrieți explicit algoritmul DDA pentru aceste puncte.

Exercițiul 2.4.11 La reprezentarea unui segment folosind algoritmul Bresenham s-a obținut parametrul de decizie inițial $p_0 = 2$. Știind că panta dreptei suport a segmentului este 0.6, calculați valoarea lui p_1 . Dați exemplu de puncte M_0 și M_{End} care să verifice aceste condiții și scrieți explicit algoritmul lui Bresenham pentru aceste puncte (indicați și valorile parametrului de decizie).

Exercițiul 2.4.12

- a) Cum se procedează în algoritmul DDA / algoritmul lui Bresenham dacă se dorește parcurgerea segmentului de la dreapta la stânga?
- b) Scrieți algoritmul DDA / algoritmul lui Bresenham pentru cazul $0 < \Delta x < \Delta y$. Ce alte situații mai pot să apară?

Exercițiul 2.4.13 Explicați cum poate fi adaptată metoda din algoritmul lui Bresenham pentru a reprezenta alte curbe (cerc, elipsă, etc.).

Exercițiul 2.4.14 Ilustrați diferența de robustețe și de eficiență pentru cele trei variante de algoritmi din secțiunea 2.2.

Exercițiul 2.4.15 Care este secvența de cod sursă care generează culoarea galben pentru o primitivă grafică?

Exercițiul 2.4.16 Câte triunghiuri vor fi desenate dacă între `glBegin(GL_TRIANGLES)` și `glEnd( )` sunt enumerate 40 de vârfuri distincte?